## Système de fichiers et gestion de fichiers

### Système de fichiers

Le **fichier** est l'unité de base du système de fichiers sous Unix/Linux. Les fichiers sont organisés sur le disque dur en **répertoires**.

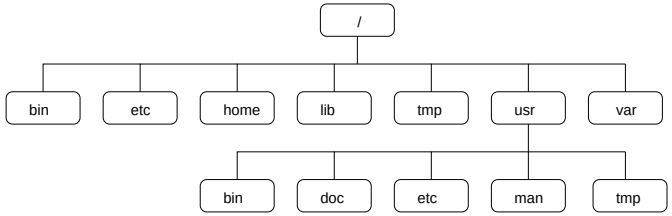
On appelle **répertoire maison** (home directory) le répertoire dans lequel se trouve l'utilisateur après l'ouverture d'une session. En général, l'utilisateur conserve ses fichiers dans ce répertoire.

On appelle **répertoire de travail** (working directory) le répertoire dans lequel travaille l'utilisateur. Toutes les commandes alors utilisées le sont dans le cadre de ce répertoire.

Deux noms de répertoires sont réservés à un usage particulier :

- **.** (un point) désigne le répertoire courant
- **..** (deux points) désigne le répertoire parent

Le **système de fichiers** Unix/Linux peut être représenté sous la forme d'un arbre traduisant la hiérarchie des répertoires et des fichiers. A la base se trouve le **répertoire racine** (root directory) désigné par /.




Les informations et les données sont réparties de la façon suivante :

- **bin** : les commandes essentielles
- **etc** : la configuration système locale
- **home** : les répertoires utilisateurs
- **lib** : les bibliothèques partagées
- **tmp** : les fichiers temporaires



### Chemins absolus et relatifs

Considérons l'arbre suivant :

Un **chemin absolu** indique les répertoires qu'il faut traverser depuis / pour arriver au répertoire ou fichier désiré :

**Exemple** : Le chemin absolu du répertoire annee2010 est :

/home/john/facture/annee2010/

Un **chemin relatif** indique les répertoires qu'il faut traverser depuis le **répertoire courant** pour arriver au répertoire ou fichier désiré. Par défaut, Linux suppose que le chemin donné est relatif.

**Exemple** : Depuis le répertoire **carole**, le chemin relatif jusqu'au répertoire **annee2010** est :

../john/facture/annee2010

**Shell**

La **shell** est l'outil le plus puissant pour interagir avec Linux.

Pour lancer la shell : **Applications-Accessoires-Terminal**.

La shell est un interpréteur de commandes. Il en existe de plusieurs types comme la sh, la csh et la bash appelées respectivement, la **Bourne shell** (du nom de son concepteur Steve Bourne), la C shell (nommée ainsi en raison de ses similitudes de syntaxe avec le langage C et finalement la **Bourne again shell** (crée par Brian Fox). Par défaut, la bash est la shell utilisée sur Ubuntu.
Ces shells présentent un prompt, traditionnellement `$`, `%` ou `#`.

La plupart des applications offrent une interface graphique plus conviviale. Cependant, les commandes données directement depuis la shell propose généralement plus d'options et permettent d'automatiser des tâches par l'intermédiaire de scripts.

Un système Unix comporte de très nombreuses commandes, par exemple une installation standard de Linux peut en compter plus de deux mille.

Certaines de ces commandes ne sont pas destinées à l'utilisateur mais au système ou à l'administrateur, cependant la grande majorité est utilisable.



**Syntaxe d'une commande**

Une commande peut-être simple et constituée par un seul mot comme la commande **date** ou alors plus complexe : d'autres informations doivent suivre la commande.

Ainsi, généralement une commande à la forme suivante :

$ **command -opt1 -opt2 ... arg1 arg2 ...**

Les règles les plus importantes régissant les commandes Unix sont :

• une commande s'écrit en minuscule

• `opt1`,`opt2`,..., désignent des options facultatives qui peuvent être spécifiées. Une option modifie le fonctionnement de la commande. Une option est généralement un suivi d'une lettre, par exemple `-a`. Lorsqu'il y a plusieurs options, on écrit `-a -b` ou `-ab`.

• `arg1`, `arg2`,..., désignent des arguments qui sont le plus souvent des noms de fichiers.

• il est obligatoire de mettre un espace entre la commande, les options et le nom de fichier

**Exemples :**

La commande **ls** affiche une liste de fichiers.

La commande **ls -l** affiche une liste de fichiers où chaque ligne décrit en détail un fichier.

La commande **ls -l chap1** affiche toutes les informations sur le fichier chap1.

**Commandes pour la gestion de fichiers/répertoires**

Conventions typographiques :
– les arguments ou options entre "<" et ">" ne peuvent pas être omis ;
– les arguments ou options entre crochets ("[" et "]") sont facultatifs.

**Répertoires et navigation**

**pwd** affiche le répertoire courant

**cd** $< dir >$ change le répertoire courant en $< dir >$

**mkdir** $< dir >$ crée un nouveau répertoire

**rmdir** $< dir >$ supprime le répertoire $< dir >$ (qui doit être vide)

**rm -r** $< dir >$ supprime le répertoire et son contenu. Attention !

**Manipulation de fichiers**

**ls** [**-alt**] liste les fichiers du répertoire courant ; options :
`-l` affichage détaillé : droits d'accès, propriétaire, date, taille
`-a` liste aussi les fichiers dont le nom commence par "`.`" (un point ; par convention, ces fichiers ne sont normalements pas listés, ils correspondent en quelque sorte aux fichiers cachés de DOS ou Windows)
`-t` trie la liste par date de création

**cp** $< f1 >< f2 >$ copie le fichier `f1` en `f2`

**mv** $< f1 >< f2 >$ renomme le fichier `f1` en `f2` ; si `f2` est un répertoire, déplace le fichier `f1` dans ce répertoire

**rm** $< f1 >$ **...** supprime les fichiers `f1` ... ; attention : cette opération est irréversible (il n'y a pas de corbeille).

**touch** $< f1 >$ crée un fichier vide nommé f1

**Remarques :**

Les commandes `cp`, `mv`, `rm`, acceptent l'option `-i` (pour *inquire*) qui demande confirmation de l'opération lorsque celle–ci altère un fichier existant (par exemple, cp f1 f2 alors que `f2` existe déjà).

**Touches spéciales pour la shell**

Les **flèches montante** et **descendante** permettent de se déplacer dans l'historique de la shell. En d'autres termes, il n'est pas nécessaire de retaper une commande, il suffit de la rechercher dans l'historique.

La touche **Tab** permet de compléter une commande ou un nom de fichier.



**Visualisation du contenu d'un fichier**

Pour accéder au contenu d'un fichier, il existe plus méthodes.
Tout d'abord des commandes pour lire uniquement le contenu du fichier : **cat** ou **more** (cat fichier ou more fichier). Le contenu du fichier s'affiche dans la shell mais il n'est pas possible de le modifier.

La commande **cat** est bien adaptée pour les petits fichiers car il déroule tout le fichier jusqu'à la fin ; il est tout de même possible de revenir en arrière avec l'ascenseur.

La commande **more** affiche le fichier écran par écran. Pour afficher la ligne suivante, tapez sur la touche Enter, pour afficher l'écran suivant, tapez sur la barre d'espace. Le pourcentage du contenu du fichier déjà affiché est indiqué à côté du prompt. Pour sortir du fichier, tapez **q**.

L'éditeur **vi** ( vi fichier) est simple, robuste et existe sur toutes les versions Unix, Linux. Il fonctionne sans interface graphique, ce qui le rend utilisable en toutes circonstances. Son fonctionnement sera décrit ultérieurement.Pour sortir du fichier :**shift ZZ**

L'éditeur **gedit** (gedit fichier) fonctionne par l'intermédiaire d'une interface graphique. Il est simple et très facile d'utilisation.

Finalement, il reste la suite burautique libre : Open Office.
Depuis la shell, **ooffice** (oofice fichier) pour utiliser l'éditeur de texte, **oocalc** (oocalc fichier) pour le tableur etc.

**Informations sur une commande**

Il existe un manuel d'utilisation disponible depuis la shell et donnant des informations sur les commandes existantes :

La commande "`man <cmde>`" où `cmde` désigne le nom d'une commande affiche à l'écran les pages de manuel correspondantes contenant la documentation complète. Pour sortir de manuel, tapez **q**.

La commande `info <cmde>` donne aussi des informations sur une commande.

La commande `whatis <cmde>` explique de façon succinte la commande.

**Wildcards**

Lorsque plusieurs fichiers ont en commun dans leur nom des caractères identiques, on utilise des wildcards pour les décrire. Les **wildcards** sont `*`, `?` ou `[,]`. Quand ils sont utilisés dans un nom de fichier donné comme argument dans une commande, leur signification est la suivante :

***** remplace un ou plusieurs caractères dans un nom de fichier.

**?** remplace un seul caractère.

[ ] entoure le choix de caractères recherchés.

**Exemples** :